

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Testing a Host's Support for Peripheral Devices

Inventor(s):

Joshua G. Poley
Jeffrey M. Alexander

ATTORNEY'S DOCKET NO. MS1-872US

TECHNICAL FIELD

This invention relates to the testing of computer-based or computer-like devices and more particularly to testing such devices' support for peripherals that communicate using a given communications protocol such as USB.

BACKGROUND

The "Universal Serial Bus" or "USB" is a standardized interconnection mechanism for connecting various types of peripherals to a computer or computer-like device. USB is perhaps most commonly thought of as a mechanism for connecting gaming peripherals such as joysticks or game controllers. However, USB is suitable for connecting many different types of peripherals to a computer, such as optical scanners, digital speakers, digital cameras, keyboards, display devices, microphones, printers, mice, and many other types of devices. The USB standard encompasses mechanical, electrical, and data format aspects involved in transferring data between a host and a plurality of peripherals.

Generally, the USB standard contemplates at least three different types of peripheral devices, referred to as Human Input Devices (HID), bulk devices, and isochronous devices. HID devices include devices that accept human input, such as keyboards, mice, joysticks, game controllers, etc. Bulk devices include mass storage devices such as hard disks, optical memory devices, and other devices that involve transferring relatively large amounts of data. Isochronous devices include speakers, microphones, and other devices with respect to which data is transferred in a time-dependent manner.

Because of the popularity of USB peripherals, it is highly desirable to provide USB support in new computer products. However, it can be a challenging

1 task to adequately test the USB support of a new computer product in light of the
2 many different types and makes of available USB peripherals. The task is
3 complicated by the very flexibility of the USB bi-directional communications
4 protocol, which allows for a nearly unlimited number of configuration and
5 communications scenarios.

6 The technology described below provides a flexible way to test peripheral
7 communications capability of a computer-like device that has been designed to
8 support a universal peripheral connection mechanism such as USB.

9 10 **SUMMARY**

11 In order to test a host's support of peripheral devices that use a particular
12 peripheral communications protocol, peripheral ports of the host are connected to
13 corresponding ports of an interface device. The interface device receives
14 peripheral messages from the in-test host using the peripheral communications
15 protocol and then packages or embeds those messages in network packets or
16 datagrams. The network packets are sent over network communications media to
17 one or more peripheral emulators, which extract or unpackage the embedded
18 peripheral messages. The peripheral emulators generate additional peripheral
19 messages, which are embedded by the emulators in network packets or datagrams
20 and sent over the network communications media back to the interface device.
21 The interface device unpackages or extracts the embedded peripheral messages
22 and sends them on to the in-test host using the peripheral communications
23 protocol. Thus, the interface device (a) receives peripheral messages from the host
24 using the peripheral communications protocol and forwards such messages to the
25 one or more peripheral emulators using a network communications protocol and

1 (b) receives peripheral messages from the one or more peripheral emulators using
2 the network communications protocol and forwards such messages to the in-test
3 host using the peripheral communications protocol.
4

5 **BRIEF DESCRIPTION OF THE DRAWINGS**

6 Fig. 1 illustrates a system for testing a host's support of peripheral devices.

7 Fig. 2 shows pertinent details of an interface device such as used in the
8 system of Fig. 1.

9 Fig. 3 illustrates data transfers that take place in the system of Fig. 1.

10 Fig. 4 shows pertinent details of a peripheral emulator such as used in the
11 system of Fig. 1.
12

13 **DETAILED DESCRIPTION**

14 Fig. 1 shows a test system 10 for testing an in-test host's support of
15 peripherals that use a given peripheral communications protocol. In the exemplary
16 embodiment described herein, the test system is designed to test support for USB
17 peripherals. However, the system can alternatively be used to test support of other
18 peripheral communications protocols.

19 The test system of Fig. 1 includes an in-test host 12. The in-test host in the
20 exemplary embodiment is a gaming device being developed for use with a wide
21 array of USB devices. In developing the host, various hardware and software
22 components (such as drivers) are being designed to support USB devices, and the
23 system of Fig. 1 is designed to exercise such hardware and software components
24 in order to test their proper operation under various conditions.
25

1 In-test host 12 might comprise one of a large number of different types of
2 computer-based or computer-like devices that operate in conjunction with various
3 peripherals. Furthermore, although input devices such as joysticks, control pads,
4 and such might be the most commonly used USB peripherals in the gaming
5 environment, the test system is designed to support other types of devices as well.
6 Specifically, the exemplary test system described herein is designed for
7 compatibility with HID, bulk, and isochronous USB devices.

8 In-test host 12 sends and receives peripheral messages using a peripheral
9 communications protocol, which in the described embodiment is the USB
10 protocol. As already described, the USB protocol encompasses mechanical,
11 electrical, and data formatting specifications. The in-test host has one or more
12 USB ports that accept USB cables for communications with peripherals using the
13 USB data transfer protocol.

14 Test system 10 further comprises one or more peripheral emulators 14 (only
15 a single peripheral emulator is used in the illustrated embodiment). In this
16 embodiment, the peripheral emulator 14 comprises a general-purpose computer.
17 Emulator 14 is programmed to emulate various peripheral devices. In the USB
18 environment, the emulator is programmed to emulate HID, bulk, and isochronous
19 USB devices.

20 The use of a general-purpose computer allows concurrent emulation of a
21 plurality of "virtual" peripherals. The peripherals are virtual in the sense that they
22 do not necessarily correspond to actual or dedicated physical hardware as would
23 normally be the case with many USB devices. It also allows testing to be
24 conducted over a wide range of normal and abnormal conditions. Thus, emulator
25 14 can be programmed to emulate normally operating peripherals as well as

peripherals that are operating abnormally, and to return normal or correct USB messages as well as USB messages that are abnormal in some sense. Furthermore, emulator 14 can be configured so that emulated devices provide the widest possible or foreseeable ranges of commands and parameters to the in-test host, and to generally simulate the most extreme expected conditions of peripheral usage. Furthermore, the emulator can be configured to emulate the connection and disconnection of peripheral devices.

Test system 10 further comprises an interface device 16. Generally, interface device 16 communicates peripheral messages with host 12 using one communications protocol, and with peripheral emulator 14 using a different protocol—a protocol that is not the same as the native protocol of the peripherals that are being emulated. More specifically, interface device 16 receives peripheral device messages from in-test host 12 using the native peripheral communications protocol (USB in the described embodiment) of the emulated peripheral devices and forwards such messages to peripheral emulator 14 using a network communications protocol (such as UDP/IP). In addition, interface device 16 receives peripheral device messages from peripheral emulator 14 using the network communications protocol and forwards such messages to in-test host 12 using the peripheral communications protocol. This process will be described in more detail below.

Interface device

Fig. 2 shows pertinent logical components of interface device 16. Interface device 16 comprises a computer having a plurality of peripheral interfaces 30 and a network interface 32. Interface device 16 further comprises operating logic 34,

1 which in the described embodiment comprises one or more processors 36,
2 associated memory 38, and one or more programs 40 that are executed by the
3 processor(s) to implement the functions of interface device 16.

4 In the described embodiment, the components of Fig. 2 are implemented in
5 a bus-based system such as an ISA bus system. Each of peripheral interfaces 30 is
6 a USB interface card that plugs into an ISA bus 42. Each card implements a
7 physical USB interface or port 43 and is configured to communicate with a
8 peripheral device port of the in-test host using the USB peripheral
9 communications protocol. USB interface cards such as this are readily available.

10 The operating logic 34 is formed by a CPU or microcontroller card that
11 similarly plugs into the ISA bus 42 and that incorporates a processor and memory.
12 The CPU card is one of a number of available CPU cards that operate from a
13 combination of volatile and non-volatile random-access memory. Typically, such
14 cards operate under a relatively simple operating system such as MS-DOS. The
15 particular card used in the described embodiment has a GPIO (general purpose
16 I/O) port 44 through which the card can be configured. Specifically, one or more
17 programs can be loaded into the non-volatile memory of the card and the card can
18 be configured to automatically initiate such programs upon startup.

19 Network interface 32 is similarly a commercially-available ISA-based card,
20 designed to support Ethernet network communications. It implements a single
21 Ethernet port 45, and is configured to communicate with emulator 14 using a
22 network communications protocol.

23 USB interfaces 30 of interface device 16 are connected by USB cables 20
24 (shown in Fig. 1) to communicate with one or more USB ports of in-test host 12
25 and to thereby communicate USB messages with in-test host 12. Interface device

1 16 is connected for communications with peripheral emulator 14 by Ethernet
2 network communications media such as a typical, Ethernet-based, local area
3 network (LAN) 22. Although the LAN shown in Fig. 1 includes only interface
4 device 16 and peripheral emulator 14, it should be understood that the LAN might
5 typically include a number of devices in addition to those shown. In other words,
6 it is not necessary to create a dedicated LAN for purposes of test system 10.
7 Rather, an existing corporate LAN can be utilized. Furthermore, any host
8 computer of the LAN can potentially be used as emulator 14.

9 In the described embodiment, network communications between interface
10 device 16 and emulator 14 are accomplished by way of UDP (user datagram
11 protocol) over IP (Internet protocol), typically referred to as UDP/IP. This
12 protocol transfers data between networked entities in the form of multiple packets
13 or datagrams. Generally, each packet or datagram contains a header and a
14 payload. The header indicates various parameters of the packet, such as its
15 destination and size, while the payload consists of the actual data that is being
16 transferred.

17 Note that although the described embodiment utilizes UDP/IP, there are
18 various other network communications protocols that could also be used. TCP
19 (transmission control protocol) over IP, typically referred to as TCP/IP, is an
20 example of a suitable alternative network communications protocol.

21 Although USB systems utilize a command response protocol, where each
22 bus has a single master that initiates all data transfers, other, non-USB systems
23 may not utilize such a command/response protocol. Thus, non-USB peripherals
24 might initiate messages, commands, or responses without receiving a previous
25 command from a master. In order to differentiate between messages that are

1 destined for the emulator 14 and those that are destined for in-test host 12,
2 messages, packets, or data ultimately destined for emulator 14 (from left to right in
3 Fig. 3) are referred to herein as "outgoing". Messages, packets, datagrams, or
4 other data ultimately destined for in-test host 12 (from right to left in Fig. 3) are
5 referred to as "incoming". This terminology is adopted strictly as a descriptive
6 tool, and is not intended to imply any particular division of responsibility or order
7 of communications between in-test controller 12 and emulator 14.

8 In general operation, the operating logic 34 of interface device 16 is
9 configured to perform actions comprising:

- 10 • Receiving outgoing peripheral device messages through peripheral
11 interfaces 30 from in-test host 12 using a peripheral communications
12 protocol.
- 13 • Sending the received outgoing peripheral device messages through
14 network interface 32 to peripheral emulator 14 using a network
15 communications protocol.
- 16 • Receiving incoming peripheral device messages through network
17 interface 32 from peripheral emulator 14 using the network
18 communications protocol.
- 19 • Sending the received incoming peripheral device messages through
20 peripheral interfaces 30 to in-test host 12 using the peripheral
21 communications protocol.

22 Because the interface device has four peripheral interfaces 30, and because
23 each interface corresponds to a single virtual device, peripheral messages can
24 pertain to any one of the four different virtual devices. Received outgoing
25 messages are distinguished based on the peripheral interface 30 that sends or

receives them. On the LAN side of interface device 16, logical network ports are used to distinguish between messages pertaining to different emulated peripherals. Specifically, UDP/IP supports the use of "sockets", which can be considered logical ports for purposes of the description herein. Assuming that emulator 14 emulates four different devices, corresponding to the four peripheral interfaces 30 of interface device 16, four different socket designations are used for outgoing peripheral device messages destined for emulator 14. Four additional different socket designations are used for incoming peripheral device messages that originate with emulator 14. Thus, with respect to a particular outgoing peripheral device message designated for a particular one of the emulated peripheral devices (as indicated by the peripheral interface 30 that received it), interface device 16 sends that particular outgoing peripheral device message to a logical port or socket corresponding to that peripheral device. Similarly, when emulator 14 sends an incoming message from a particular emulated device, to be provided to in-test host 12 from a particular peripheral interface 30, the message is sent to a logical port or socket on interface device 16 corresponding to that peripheral device. In the described embodiment, UDP ports 201-204 are used for outgoing messages (received by emulator 14), and UDP ports 211-214 are used for incoming messages (sent from emulator 14). The operating logic of interface device 16 maintains a correspondence between logical ports and peripheral interfaces 30.

Generally, interface device 16 is designed and configured to simply pass command messages back and forth between in-test device 12 and emulator 14, without involvement in the actual data content of the messages. However, it may be desirable in certain situations for the interface device 16 to respond to messages from in-test device 12 without waiting from a response from emulator 14. In the

described embodiment, for example, interface device 16 is configured to automatically return acknowledgement messages (ACKs) to in-test device 12 at appropriate intervals while waiting for a command response from emulator 14.

In addition, interface device 16 is configured to process USB checksums (CRCs) that form part of each USB message. Specifically, interface device 16 verifies the checksum of each USB command message received from in-test device 12 and adds checksums to USB response messages before sending them to in-test host 12.

Data Flow

Fig. 3 illustrates data flow through interface device 16 in the specific context of a USB system. In the USB environment, in-test device 12 is assumed to be a bus master. Each of USB interface cards 30 is considered to be a single device. Communications are initiated by USB command messages issued from in-test device 12. Peripheral devices (emulated within peripheral controller 14) respond to commands with USB response messages. Thus, in a USB system, “command messages” are issued by in-test host 12 and are eventually destined for emulator 14. “Response messages” are issued by emulator 14 and are eventually destined for in-test host 12. The USB messages potentially include HID, bulk, and isochronous USB messages.

Fig. 3 shows a single transaction in which in-test host 12 generates a USB command message and peripheral emulator 14 generates a response message. The sequence involves four data transfers, labeled as follows in Fig. 3: (1) USB Command Message; (2) UDP/IP Command Packet; (3) UDP/IP Response Packet; and (4) USB Response Message. The transfers take place in the order listed.

1 Operating logic 34 is configured to perform these transfers. More specifically, one
2 or more of programs 40 comprise instructions that are executed by processor(s) 36
3 to perform these transfers.

4 (1) USB Command Message. A USB command message is initially sent or
5 transmitted by in-test host 12 to a USB interface card of interface device 16 over
6 the USB communications media (a USB cable). The command message is
7 formatted in accordance with the USB standard. The USB command message is
8 received from in-test host 12 by interface device 16 through one of its interface
9 cards 30.

10 (2) UDP/IP Command Packet. Interface device 16 packages or embeds the
11 received command message within the payload of a corresponding data packet
12 formatted in accordance with a network communications protocol—within a
13 UDP/IP packet or datagram. Interface device 16 then sends or transmits the
14 UDP/IP packet (containing the embedded USB command message) to emulator 14
15 via LAN 22, using the UDP/IP network communications protocol. Emulator 14
16 receives the UDP/IP packet and its embedded USB command message from
17 interface device 16 and unpackages or extracts the embedded USB command
18 message.

19 (3) UDP/IP Response Packet. An emulated or virtual device within
20 emulator 14 receives the extracted USB command message and formulates an
21 appropriate USB response message in accordance with emulated device
22 characteristics and desired test scenarios. The response message is initially
23 formatted in accordance with the USB protocol, and is then embedded or
24 packaged within a UDP/IP packet or datagram. The emulator then sends the
25 UDP/IP packet or datagram, containing the USB response message, back over

1 LAN 22 to interface device 16 using the UDP/IP network communications
2 protocol. Interface device 16 receives the UDP/IP response packet and its
3 embedded USB response message and unpackages or extracts the embedded USB
4 response message.

5 (4) USB Response Message. The extracted USB response message is then
6 sent or transmitted through one of USB interfaces 30 over the USB bus or cable to
7 in-test host 12, using the USB communications protocol. In-test host 12 receives
8 the USB response as if it is from an actual, physical device.

9 10 **Peripheral Emulator**

11 Fig. 4 shows pertinent components of general purpose computer 14, which
12 is used as a peripheral emulator to emulate one or more peripheral components.
13 Generally, various different general purpose or special purpose computing system
14 configurations can be used for emulator 14, including but not limited to personal
15 computers, server computers, hand-held or laptop devices, multiprocessor systems,
16 microprocessor-based systems, programmable consumer electronics, network PCs,
17 minicomputers, mainframe computers, distributed computing environments that
18 include any of the above systems or devices, and the like.

19 The functionality of computer 14 is implemented largely by means of
20 computer-executable instructions, such as program modules, that are executed by
21 the computer. Generally, program modules include routines, programs, objects,
22 components, data structures, etc. that perform particular tasks or implement
23 particular abstract data types. Tasks might also be performed by remote
24 processing devices that are linked through a communications network. In a
25

1 distributed computing environment, program modules may be located in both local
2 and remote computer storage media.

3 The instructions and/or program modules are stored at different times in the
4 various computer-readable media that are either part of the computer or that can be
5 read by the computer. Programs are typically distributed, for example, on floppy
6 disks, CD-ROMs, DVD, or some form of communication media such as a
7 modulated signal. From there, they are installed or loaded into the secondary
8 memory of a computer such as magnetic-based hard disks. At execution, they are
9 loaded at least partially into the computer's primary electronic memory. The
10 invention described herein includes these and other various types of computer-
11 readable media when such media contain instructions, programs, and/or modules
12 for implementing the actions described below in conjunction with a
13 microprocessor or other data processors. The invention also includes the computer
14 itself when programmed according to the methods and techniques described
15 below.

16 For purposes of illustration, programs and other executable program
17 components such as the operating system are illustrated herein as discrete blocks,
18 although it is recognized that such programs and components reside at various
19 times in different storage components of the computer, and are executed by the
20 data processor(s) of the computer.

21 With specific reference to Fig. 4, computer 14 has operating logic 62 that
22 comprises one or more processors 63, memory 64, and one or more programs.
23 The pertinent programs in this example comprise an operating system 65 and
24 emulation programs 66. Computer 14 also includes various I/O and user interface
25

1 components 67 such as a keyboard, visual display, etc. that are not specifically
2 shown in Fig. 4.

3 Computer 14 further comprises a network interface 68 for communications
4 with LAN 22 and interface device 16 (Fig. 2). In this example, the computer is
5 configured to communicate with interface device 16 and potentially with other
6 networked components using a network communications protocol such as IP.
7 Specifically, UDP/IP is used to communicate with interface device 16. Although
8 the example described herein utilizes Ethernet network communications media,
9 other forms of network communications media and protocols might alternatively
10 be used.

11 Generally, emulation programs 66 are designed to respond to peripheral
12 command messages received from interface device 16 and to create response
13 messages that exercise the functionality of in-test host 12. More specifically, the
14 emulation programs might comprise four different emulation programs, each
15 corresponding to a different emulated or virtual USB device. Each such emulation
16 program receives network data packets designating a particular respective logical
17 port or socket, such as UDP sockets 201-204 as mentioned above. The network
18 data packets have embedded USB command messages. Each emulation program
19 responds to the embedded USB command messages by creating USB response
20 messages which are in turn embedded in network data packets and sent back to
21 interface device 16 using corresponding logical ports or sockets, such as UDP
22 sockets 211-214 as mentioned above.

23 More specifically, the emulation programs are configured to perform
24 actions comprising:
25

- Receiving data packets over a network communications media, wherein the data packets are formatted in accordance with a non-USB network communications protocol such as UDP/IP (with embedded USB command messages);
- Unpackaging USB command messages from the received data packets;
- Emulating one or more USB peripherals that respond to and create USB messages in order to create USB response messages;
- Packaging the USB response messages in data packets formatted in accordance with the non-USB network communications protocol; and
- Sending the data packets over the network communications media using the non-USB network communications protocol.

Interface device/Emulator Data Format

Data communications between interface device 16 and emulator 14 are formatted in accordance with the UDP/IP communications protocol. Each data packet contains a payload formatted in a manner that allows for efficient packaging and unpackaging of USB data. Although the particular format of the payload will vary from one embodiment to another, the embodiment described herein uses the following fields within each UDP/IP payload transferred between interface device 16 and emulator 14:

- Message Type: a two byte code indicating a message type and an optional sub-type.

- USB Endpoint: a one byte value indicating with which USB endpoint the message is associated.
- Data Size: a two-byte value indicating the size of a data block that follows.
- Data: a field of variable length as indicated by the “Data Size” field.

The most frequently used message type is one in which the “Data” field contains USB data for transfer between in-test device 12 and emulator 14. Another type of UDP/IP message is defined for handshaking messages such as ACKs and NAKs between interface device 16 and emulator 14.

As described above, each UDP/IP data packet is directed to or received from a logical port or socket that corresponds to a particular one of peripheral interfaces 30, each of which represents a physical device. The USB protocol includes the concept of “endpoints” within each physical device. The “USB Endpoint” field is used in USB data transfers to indicate which endpoint the data is associated with. Note that this field is ignored in messages that are not associated with a particular endpoint.

Other UDP/IP message types apply to status and configuration messages between interface device 16 and emulator 14. These types allow emulator 14 to query interface device 16 for status information, version information, logical address information, etc. In addition, these message types allow emulator 14 to set various parameters within interface device 16. For example, in the USB environment each of the USB peripheral interfaces 30 might have settings that are desirably set by emulator 14 depending on the nature of the USB device that is being emulated. As another example, it might be desirable to allow emulator 14 to set the USB address of each interface 30. Furthermore, these message types might

1 allow emulator 14 to perform functions such as instructing a peripheral interface
2 30 to connect to or disconnect from in-test device 12 , or to instruct interface
3 device 16 to reset itself.

4 For messages not associated with a particular peripheral interface 30,
5 interface device 16 listens on UDP socket 200. Furthermore, interface device 16
6 responds to UDP broadcast messages on UDP socket 200. This capability allows
7 peripheral emulator 14 to detect the presence and IP address of interface device 16
8 on LAN 22.

9 10 **Conclusion**

11 Although details of specific implementations and embodiments are
12 described above, such details are intended to satisfy statutory disclosure
13 obligations rather than to limit the scope of the following claims. Thus, the
14 invention as defined by the claims is not limited to the specific features described
15 above. Rather, the invention is claimed in any of its forms or modifications that
16 fall within the proper scope of the appended claims, appropriately interpreted in
17 accordance with the doctrine of equivalents.